3600 SOFTWARE GUIDE

INTRODUCTION

revised 10/19/83

The 3600 is a product which combines the ATARI 2600 hardware with a new graphics chip called MARIA. The entire 2600 library of cartridges will run on the 3600 as they do on the 2600, but new cartridges designed to access the improved hardware will be able to take advantage of a large number of improvements.

OVERVIEW OF 3600

Ignoring the 2600 environment, which is identical to the ATARI 2600, the 3600 environment is characterized by the following:

```
(2) 6116's      - 4K bytes of RAM.
    6532        - I/O.
    TIA         - sounds, some input ports.
    Expanded cartridge slot.
    SALLY(6502)- microprocessor running at 1.79 MHz.
    MARIA       - all video.
```

Additionally, there is a protection circuit which verifies that each cartridge has the correct encrypted data before enabling 3600 mode. Encryption will be covered in another document, but see Appendix 1, 3600 Memory Map, for information about reserving space for encryption.

6116's

There are two (2) 6116 2Kx8 RAM chips on the 3600 PC board. Together they occupy addresses x'1800' to x'27FF'. They are also partly accessible (shadowed) at addresses x'0040' - x'00FF' and x'0140' - x'01FF' to extend zero page (quick access) RAM and first page (stack) RAM. Refer to the memory map appendix for further information.

6532

This chip is used only for I/O in 3600 mode, whereas in 2600 mode it also supplies all RAM and timers. Its functions are more limited because its speed is not fast enough for normal operation. Any access to this chip (joystick, and switch I/O) will cause the microprocessor to slow to 1.19 MHz. The ports and switches connected through the 6532 are: joysticks (directional), pause, game select, game reset, and difficulty switches. The 6532 can be used to generate output through the joystick ports as well. For address information on 6532 ports and switches, refer to Appendix 2, Standard 3600 Equates.

TIA

The TIA is only partly accessible in 3600 mode. While it occupies addresses x'0000' - x'003F' in 2600 mode, only the section at

x'0000' - x'001F' is available in 3600 mode. The only significant (useable) registers of these are the sound related registers and the input ports (fire buttons, paddle controllers). Any access to the TIA will cause the processor to slow from 1.79 MHz to 1.19 MHz.

CARTRIDGE SLOT

The cartridge slot is larger for 3600 mode cartridges. The additional lines are: three (3) address lines (now all 16 address lines appear on the cartridge connector); the READ/WRITE line, so that RAM may be added to any cartridge very simply; the phase 2 clock line in order to add another microprocessor on the cartridge and have it synchronized with the existing Sally chip; an audio line so that one may mix in audio signals generated on the cartridge; a composite video line, so that external video signals may be included; and the HALT line, to enable the cartridge to distinguish MARIA ROM accesses from SALLY ROM accesses.

SALLY (6502)

This is the microprocessor, which is also used in the ATARI 5200. The only thing special about the Sally chip is that it has a HALT line, which allows the functionality described above.

MARIA

This is the custom chip which is the heart of the 3600. It handles all graphics and video including the VSYNC and VBLANK signals.

OVERVIEW OF MARIA

GRAPHICS

MARIA does not employ the concepts of players, missiles, and playfield, as do the 2600 and 5200. Instead MARIA uses an approach to graphics commonly used in coin-operated games. Each raster of the display may be thought of as a bit map. This map is contained in an area of the MARIA chip called the Line RAM. Information is first stored into the Line RAM, then later read from Line RAM and displayed on the screen.

Consider for a moment just one raster of display. One would compose this raster's graphics by storing data into Line RAM. This is done by specifying what data should be put at what horizontal location. Graphics may be specified in small pieces, and overlapped. The order in which pieces of a raster are specified determines object priority with the last object specified on top.

When graphic data is specified to be stored into Line RAM, it will reference any one of eight (8) color palettes. Each pixel of data will take on any one of three (3) colors from the specified palette, or may be turned off (transparent). Again, the Line RAM contains only one raster of graphics information. There are actually two Line RAM

buffers. While one is being read (displayed), the other is being written for display the next raster. This means that the construction of graphics for a raster may take as long as, but no longer than, one raster, and that graphics must be stored into Line RAM on a raster by raster basis.

The only limit to the number, and size of objects on one scan line is the amount of time it takes to load each into Line RAM, as all loading must occur during one scan line.

DISPLAY

There are a total of 262 rasters per frame (1/60th second). The "visible" screen (during which MARIA attempts display) starts on raster 16 and ends on raster 258. The area found visible on all television sets starts on raster 41 and ends on raster 232, 192 scan lines later. Any display outside this area may not appear on all televisions. See Appendix 4, Frame Timing, for more details.

Display is accomplished automatically by MARIA and consists of two tasks: constructing the Line RAM, and displaying the graphics. These happen simultaneously in MARIA. Construction of Line RAM is automatically initiated every raster by MARIA, and is directed by a predefined list of instructions called the Display List. Line RAM construction occurs through a process called DMA (Direct Memory Access). This means that the 6502 (SALLY) processing is suspended while MARIA comes in and interrogates the RAM and ROM for Display List and graphics information. DMA will occur every "visible" scan line and lasts no longer than one scan line. Because the Line RAM being constructed is displayed on the following scan line, MARIA will read each Display List one line before it is actually displayed. All Line RAM is cleared on a line by line basis and BACKGRND color will be displayed if no data is written.

Display List

DMA is mainly concerned with reading the Display List. This is a list of instructions for where to find graphics data, where to put it on the screen, and other details for constructing a scan line. The Display List is made up of many "headers." Most headers are four (4) bytes long (the exception is discussed later). If the second byte of a header is zero, it indicates the end of the Display List, and DMA will stop, allowing the 6502 to continue processing. The format of the header is as follows:

A7	A6	A5	A4	A3	A2	A1	A0
P2	P1	P0	W4	W3	W2	W1	W0
A15	A14	A13	A12	A11	A10	A9	A8
H7	H6	H5	H4	H3	H2	H1	H0

or

LOW ADDRESS							
PALETTE			WIDTH				
HIGH ADDRESS							
HORIZONTAL POSITION							

where:

ADDRESS (A15-A0) - Address of graphics information.
PALETTE (P2-P0) - Refers to color palettes 0-7.
WIDTH (W4-W0) - 2's complement of width.
Specifies number of bytes of graphics data to fetch: values 1-31.
HORIZONTAL POSITION (H7-H0) - X location on the screen where left edge of graphics is to be placed.
0-159 => Visible.
160-255 => Not visible.
Wrap around occurs at 255/0 boundary.

Each header is concerned with one graphics item, which can be any width. If ten objects should appear on a scan line, the Display List for that scan line would be ten (10) headers long, followed by two (2) bytes, the first of which is ignored, and the second of which should be zero to end DMA.

A Display List may cross only one page boundary, so it can be no more than 512 bytes long. Additionally, Display Lists must be in RAM, due to the required access time.

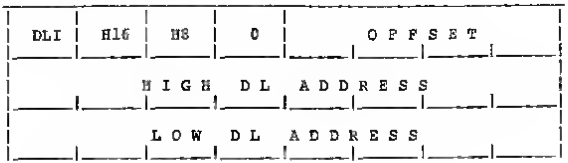
Display List List

MARIA locates the Display Lists by reading a Display List List (referred to as DLL from now on). This list is a series of 3 byte entries. Each entry points to a Display List. Included in each entry is a value called OFFSET, which indicates how many rasters should use the specified Display List. OFFSET is decremented at the end of each raster until it becomes negative, which indicates that the next DLL entry should now be read and used. Each time graphics data is to be fetched, OFFSET is added to the specified High address byte, to determine the actual address where the data should be found. This allows one display list to specify many rasters of graphics. Without OFFSET the only approach to graphics is to have a Display List for each raster, and a DLL for each Display List. Not only would this use a lot of RAM, but it would also take quite a bit of processing time to manipulate these Display Lists when objects move. Because OFFSET is added to the HIGH address byte, each raster of graphics for an object must be separated by x'100' bytes, or one page.

The group of rasters specified by one DLL entry is called a "zone." Again, the number of rasters in a zone equals OFFSET+1. Larger zones mean less RAM is needed for DLLs, Display Lists, and Character Maps (see DMA MODES below). But upon consideration of how to use zones, you will realize that to achieve smooth vertical motion each stamp must be padded at top and bottom with zeros. For example, if the top raster of an object is to appear on the last line of a 16 high zone, it must have 15 lines of zeros above it. If that object is 8 pixels (2 bytes) wide, and its top line of data is located at x'CF04' and x'CF04', then you will need two bytes of zeros at x'D004', x'D104', x'D204', x'D304', ... , and x'DE04' (remember that OFFSET decrements). As this can add up to many pages of zeros, you can specify that MARIA should interpret certain data as zeros, even if it isn't. This is called "Holey DMA" because DMA will see "holes" in the data that aren't really there. This can be enabled and disabled on a zone by zone basis via a DLL entry. Holey DMA has been aimed at 8 or 16 raster zones, but will have the same effect for other zone sizes. MARIA can be told to interpret odd 4K blocks as zeros, for 16 high zones, or odd 2K blocks as zeros for 8 high zones. This will only work for addresses above x'8000'. This means that these blocks can hold meaningful code, or tables, or graphics data used in a zone where Holey DMA is not on.

One of the bits of a DLL entry tells MARIA to generate a Display List Interrupt (DLI) for that zone. The interrupt will actually occur following DMA on the last line of the PREVIOUS zone. This interrupt is non-maskable, and causes the processor to go to the address specified by the NMI vector at x'FFFA' and x'FFFB'. This interrupt in no way affects DMA, so processing will still be suspended at the beginning of the next raster.

The format of a 3 byte DLL entry is as follows:



where:

DLI	-	Display List Interrupt flag. 0 => No DLI. 1 => Interrupt after DMA on last line of previous zone.
H16	-	16 high zone Holey DMA enable. 0 => Not enabled. 1 => Enabled. DMA interprets odd 4K blocks as zeros. (A12 high => data=0)
H8	-	8 high zone Holey DMA enable. 0 => Not enabled. 1 => Enabled. DMA interprets odd 2K blocks as zeros. (All high => data=0)
OFFSET	-	OFFSET starting value. 4 bits only.
DL ADDRESS	-	Address of Display List for this zone.

A Display List List may cross only one page boundary, so it can be no more than 512 bytes long. Additionally, Display List Lists must be in RAM, due to the required access time.

MODES

DMA Modes

There are two modes for specifying graphics data. The first, called Direct mode, is what has just been explained, where a Header (in the Display List) points directly to graphics data. The other mode is called Indirect or Character mode, and is somewhat different in that the Header points to a Character Map, which in turn points to graphics data. Indirect mode is selected by every header that requires it via an extended (5 byte long) header. The format of this header is as follows:

A7	A6	A5	A4	A3	A2	A1	A0
WM	1	IND	0	0	0	0	0
A15	A14	A13	A12	A11	A10	A9	A8
P2	P1	P0	W4	W3	W2	W1	W0
H7	H6	H5	H4	H3	H2	H1	H0

or

LOW ADDRESS							
MODE BYTE							
HIGH ADDRESS							
PALETTE			WIDTH				
HORIZONTAL POSITION							

where:

ADDRESS (A15-A0) - Address of graphics information.
MODE BYTE: WM - Write mode bit.
0 => 160x2 or 320x1
1 => 160x4 or 320x2
IND - 0 => Direct mode.
1 => Indirect mode.
PALETTE (P2-P0) - Refers to color palettes 0-7.
WIDTH (W4-W0) - 2's complement of width.
Specifies number of bytes of graphics data to fetch: values 1-32.
HORIZONTAL POSITION (H7-H0) - X location on the screen where left edge of graphics is to be placed.
0-159 => Visible.
160-255 => Not visible.
Wrap around occurs at 255/0 boundary.

There is an added bonus to five byte headers. Because the end of DMA is indicated by the presence of a zero in the second byte of a header, and in a five byte header the width byte is not the second but the fourth, a width of zero is valid in an extended header, and will be interpreted as a value of 32.

Indirect mode, when selected, only lasts as long as the corresponding header is being processed. MARIA will return to Direct mode before the next header is read.

In indirect mode, the width indicates how many Character Map references to make, where each Character Map entry points to one byte of graphics data (the Character Map can point to two (2) consecutive bytes of graphics; see CTRL under REGISTERS). The idea behind Character (Indirect) mode is to is to specify a great amount of graphics with only one Header. The graphics start at the horizontal location specified by the Header and each character (graphics referred to by one Character Map entry) is inserted to the right of the previous one. One Character may be changed without affecting the others by altering the Character Map entry corresponding to that character. This is ideally suited for backgrounds such as the maze and dots in Ms. Pacman.

The Character Map is composed of W entries, where W is the specified width and each entry is one byte long. Each entry is a Low address byte of a character, and the High address byte is specified by the Character Base register (see CHARBASE under REGISTERS). This means that each character on a scan line must have the same high address byte (sit on the same 256 byte page).

Display Modes

The normal display mode is 160 mode, where the screen is divided into 160 pixels horizontally. Typically graphics are done in 160x2 mode, where there are two color bits specified for each pixel, and these two color bits refer to one of the eight palettes. Alternately, one may specify graphics in 160x4 mode, where there are four color bits per pixel. In this mode, each byte of graphics data would specify only two (2) pixels of graphics. If higher resolution is preferred, 320x1 mode is the common choice, where the screen is divided into 320 pixels horizontally and each pixel has one color bit. A more colorful 320x2 mode is also available with two color bits per pixel.

Selection of a particular mode is accomplished through two separate operations: specification of WRITE MODE, and specification of READ MODE. WRITE MODE is specified via the WM bit of an extended (5 byte) header, as described above. READ MODE is specified via the CTRL register. Both of these specifications will remain in effect until re-specified. WRITE MODE is not initialized by MARIA on power-up, and must be initialized by the cartridge before any display occurs. The reason for specifying WRITE MODE via an extended header, is to allow the programmer to change from 160x2 to I60x4 (or from 320x2 to 320xI,

or vice-versa) during the DMA for a particular scan line. For more information about modes see CTRL under REGISTERS.

REGISTERS

The location of the MARIA registers which control the display is shown in Appendix 1, 3600 Memory Map.

Palettes

The palette registers are used to specify colors for the graphics. There are eight palettes, and each contains three colors. The colors themselves are specified in the form:

C3	C2	C1	C0	L3	L2	L1	L0

where C3-C0 is the color, and L3-L0 is the luminosity, for a total of 256 different hues.

The palette registers are labeled P0C1, P0C2, P0C3, P1C1, P1C2, P1C3, P2C1, P2C2, P2C3, ... P7C1, P7C2, P7C3. A pixel whose two color bits are "10" and which refers to palette three (3) would be colored based on the value in P3C2. Color zero of any palette is transparent. Additionally, there is a register called BACKGRND used to specify background color. All the palettes and BACKGRND are READ/WRITE, but they must be read using "Absolute,Index" addressing of the 6502.

OFFSET

The OFFSET register is a 4 bit value which gets added, automatically, to the high address byte on any graphics data fetch, whether Direct or Indirect. This register is internal to MARIA, and is set by each Display List List entry.

In a previous incarnation, the OFFSET register occupied a memory address. This address is now vacant, but you should STORE ZERO THERE ON POWER-UP TO ALLOW FOR FUTURE EXPANSION.

CHARBASE

The CHARBASE register serves to specify the high address for any graphics data fetch in Character (Indirect) mode. As you recall, the Character Map (pointed to by the Header in the Display List) specifies the low address bytes of graphics data. Each of these low address bytes is concatenated with the sum of CHARBASE + OFFSET, to give the full 16 bit addresses of where the graphics data should be found. The CHARBASE register is WRITE ONLY.

DPPH

DPPH stands for Display List Pointer Pointer High, and this is the register which contains the high address byte of the Display List List. This register is WRITE ONLY. The Display List List may cross one page boundary, in which case DPPH is internally incremented, then reset at the end of the visible screen, so it is valid for the next frame. This register (and DPPL) should be written to before DMA is turned on. Once DMA is on, DPPH and DPPL may be written at any time, as they are only read at the beginning of the screen.

DPPL

This register is used to specify the low address byte of the Display List List. It, too, is WRITE ONLY.

MSTAT

MSTAT is a READ ONLY register which communicates the status of Vertical Blank via bit 7 (MSB). When this bit is 1 VBLANK is on. When VBLANK turns off, DMA will begin according to your Display List. This transition occurs at raster 16 of the frame.

CTRL

The CTRL register is a WRITE ONLY register used to control many of the modes of MARIA. Through this register one can control whether the background color extends off the edge of the TV (horizontally), beyond the area where graphics may be positioned; or whether the background color stops at the horizontal limits of graphics and this border area appears black. This border area is an area which appears undependably on various television sets.

CTRL also specifies whether characters (in Character mode) are one or two bytes wide. That is, in Character (Indirect) mode, whether one, or two bytes of graphics data should be fetched at the address pointed to by the Character Map entry and CHARBASE. The advantage of two byte characters is that the same number of pixels can be specified with half as many Character Map entries. The disadvantage is that when changing one character, twice as much of the screen is affected.

This register also controls whether the color burst signal is generated or not. If color burst is turned off, the graphics are, of course, displayed in black and white, but with a greater clarity than if the gray scale colors (x'00' - x'0F') were used.

Another bit of CTRL enables "Kangaroo" mode which eliminates transparency, so that any pixel of color "0" will be background color, rather than transparent. For the derivation of this name see the ATARI coin-op game Kangaroo.

DMA may be turned on or off via the CTRL register. At power-up DMA is off, and must be turned on by the cartridge. This should not be done until after DPPL and DPPH have been stored (so that DMA doesn't

try to read a DLL from an undefined location). DMA should be turned on DURING VBLANK, and never during the screen (rasters 16-258). If DMA is off the screen will continue to display the background color.

Finally, CTRL is where the READ MODE portion of the graphics mode is selected (remember the WRITE MODE portion is selected via an extended header). WRITE mode controls the way data is written into Line RAM, and READ mode controls the way Line RAM is interpreted and translated to the screen. Because READ MODE affects the scan line being displayed, changes to READ MODE should happen at the beginning of the scan line to be affected.

The WRITE MODE selects between a.) 160x2 or 320x1 and b.) 160x4 or 320x2. The Read mode selects between a.) 320A or 320C, b.) 320B or 320D, and c.) 160A(x2) or 160B(x4). The following table should be more informative:

MODE	WM	RM1	RM0
160A	0	0	0
160B	1	0	0
320A	0	1	1
320B	1	1	0
320C	1	1	1
320D	0	1	0

320A mode is a true 320x1 mode. Pixels that are "on" refer to color two (2) of the specified palette. Pixels that are off are transparent (or background color if "Kangaroo" mode is on. In 320B mode, which is a 320x2 display mode, only the most significant palette bit is read. This means that either palette zero (0) or palette four (4) is used. If "Kangaroo" mode is off, transparency will work differently for 320 modes. Consider a pair of 320-size pixels which make up one 160-size pixel. If either pixel of the pair is off, it will not be transparent, but will take on background color instead. If both pixels are off, they will be transparent. With "Kangaroo" mode on, things work as one would expect them to work in this mode. Another factor concerning 320 modes is that the horizontal positioning still happens like 160 mode. This means that in 320 modes, objects can only be positioned in 2 pixel increments.

320C and 320D are display modes somewhat similar to 320B and 320A, respectively. They are what you would get if you changed WRITE mode without changing READ mode (such as changing modes during a scan line). If you were in 320A mode, and wanted to include a character with more colors on the line, changing modes would give you 320C mode. Likewise, changing from 320B on the fly would give you 320D mode. The way data is interpreted for 320C and 320D will be explained later on.

In 160x4 mode, again only the most significant palette bit is read (note that 160x4 and 320B share the same WRITE mode sense). Because there are more color bits than each palette can handle, the palettes are combined in 160x4 mode so you may choose between the combinations of 0-3 and 4-7. The net result of 160x4 mode is twelve

(12) colors, where color one (1) is P0C1 or P4C1, two (2) is P0C2 or P4C2, five (5) is P1C1 or P5C1, six (6) is P1C2 or P5C2, etc. and colors 0,4,8, and 12 are transparent.

The CTRL rsgister is arranged as follows:

CK	DM1	DM0	CW	BC	KM	RM1	RM0

where:

CK - Color Kill.
0 => Normal color.
1 => No color burst.

DM1,DM0 - DMA control.
0 => Test A (DO NOT OSE).
1 => Test B (DO NOT OSE).
2 => Normal DMA.
3 => No DMA.

CW - Character Width.
0 => Two (2) byte characters.
1 => Single byte characters.

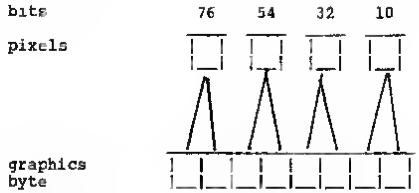
BC - Border Control.
0 => Background color border.
1 => Black border.

KM - "Kangaroo" Mode Switch.
0 => Transparency.
1 => "Kangaroo" mode: no transparency.

RM1,RM0 - Read Mode.
0 => 160x2, or 160x4
1 => Not used.
2 => 320B or 320D.
3 => 320A or 320C.

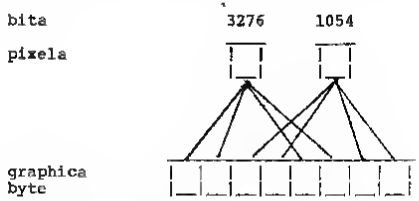
(WARNING: TEST A (DM = 0) and TEST B (DM = 1) should NOT be used! These are for testing the chip at manufacturing time, and may cause irrecoverable problems, as well as possible DAMAGE TO THE BASE UNIT!)

The coding of graphics data is straightforward for most of these modes. In 160x2 mode, each pair of bits is arranged so that the leftmost pixel's color is specified by the most significant pair of bits, and the rightmost pixel by the least significant pair of bits.



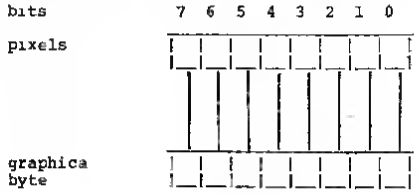
160x2

In 160x4 mode, the data is read as follows: the left pixel's color is specified by bits 3, 2, 7, 6 (where 3 is MSB, 6 is LSB). The right pixel is specified by bits 1, 0, 5, 4 (where 1 is MSB, 4 is LSB).



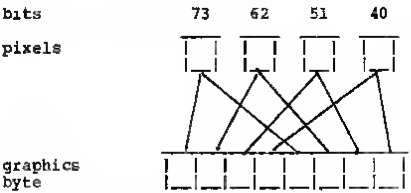
160x4

320A mode is a direct mapping, like 160x2, except that each bit specifies the color of one pixel.

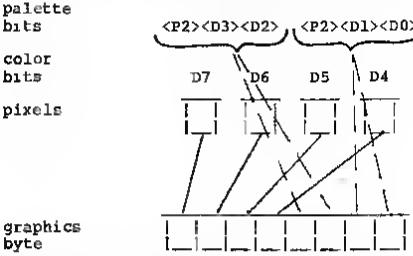


320A(x1)

320B mode works as follows:

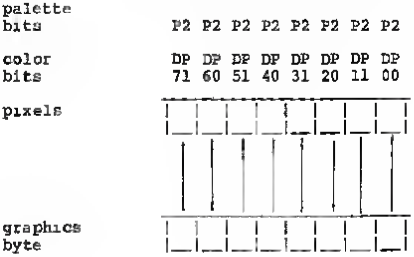


320C mode allows more colors than 320A, but cannot really be called 320x2. In this mode, some of the graphics data goes to specifying palettes, which is somewhat non-standard. If a pixel is on, it is color two (2), and if it is off, it is transparent, or background color (same as 320A and 320B). The palette is determined by combining the most significant palette bit with two data bits, so each byte of graphics can refer to a different palette. The palette for the leftmost pixel is specified by P2, D3, and D2 (where P means a palette bit, and D means graphics data bit), and the graphics are specified by D7. The next pixel right uses the same palette, and uses D6 for data. The next pixel right uses a palette specified by P2, D1, and D0, and uses D5 for data. And the rightmost pixel uses the same palette, but D4 for data. The mapping for 320C mode is as follows:



320D mode is a little confusing, too. Every pixel refers to the same palette but palette bits affect the color of the pixels. The only palette bit used for palette definition is the most significant bit (same as 320B), so only palettes zero (0) and four (4) will be referenced. For color selection there is really more than one bit per pixel. The graphics data bits are used as follows: each is the most significant bit of a two bit pair. But the least significant bit of this pair is either P0 or P1 (where P again means palette bit). If the specified palette is 0 or 4 (where P1 and P0 are zero), this is a normal 320x1 mode, like 320A. But if the specified palette is 5,

palette 4 will be used, and certain pixels will be either color 1 or 3, and others will be 0 or 2. A picture's worth a thousand words, so:



320D

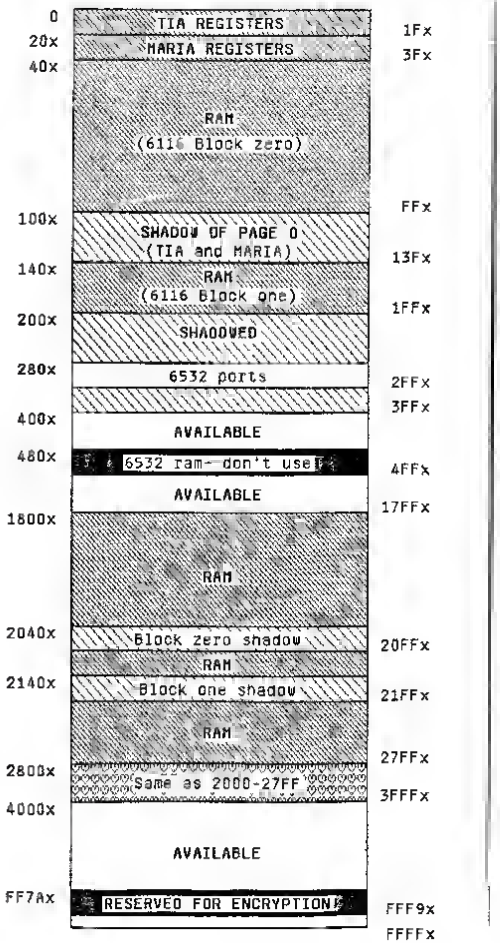
APPENDIX 1: 3600 MEMORY MAP

The memory map of the 3600, graphically illustrated on the next page, is in many ways similar to that of the 2600, with the addition not only of MARIA, but also of 4K of RAM. This RAM is shadowed (responds to other addresses) in zero, first, second, and third pages, the first two of these being significant. You will notice the absence of the 128 bytes of 6532 RAM that make up zero page RAM in the 2600. This is because of a speed discrepancy with the 6532. It's RAM has moved to an area in page four (4) and may not exist in future versions of the MARIA chip, so it should not be used.

		FROM		TO
1.	TIA	0000 00XX 0000 0000	-	0000 00XX 0001 1111
2.	MARIA	0000 00XX 0010 0000	-	0000 00XX 0011 1111
3.	6532 PORTS	0000 0010 1000 0000	-	0000 0010 1111 1111
4.	6532 RAM (DON'T USE)	0000 010X 1000 0000	-	0000 010X 1111 1111
5.	RAM	0001 1000 0000 0000	-	0010 0111 1111 1111
6.	RAM SHADOW	00X0 000A 0100 0000	-	00X0 000A 1111 1111
7.	RAM SHADOW	001X X000 0000 0000	-	001X X111 1111 1111

where: X means "Don't Care," and A means the bits may be 1 or 0, but are not ignored. Entries 5 and 6 indicate that pieces of RAM from x'1800' - x'27FF' appear in zero, and first pages. The last entry indicates that the last 2K block (x'2000' - x'27FF') is repeated at x'2800', x'3000', and x'3800' making this 6K area a series of 2K shadows.

For encryption purposes, the 128 bytes from x'FF7A' - x'FFF9' must be left free. Put zeros in this area until encrypted.



APPENDIX 2: STANDARD 3600 EQUATES

```
INPTCTRL    EQU X'01'   INPUT PORT CONTROL ("VBLANK" IN TIA)     WO
AUDCU       EQU X'15'   AUDIO CONTROL CHANNEL 0                 WO
AUDC1       EQU X'16'   AUDIO CONTROL CHANNEL 1                 WO
AUDF0       EQU X'17'   AUDIO FREQUENCY CHANNEL 0               WO
AUDF1       EQU X'18'   AUDIO FREQUENCY CRANNEL 1               WO
AUDV0       EQU X'19'   AUDIO VOLUME CHANNEL 0                  WO
AUDV1       EQU X'1A'   AUDIO VOLUME CHANNEL 1                  WO
INPT0       EQU X'08'   PADDLE CONTROL INPUT 0                  WO
INPT1       EQU X'09'   PADDLE CONTROL INPUT 1                  WO
INPT2       EQU X'0A'   PADDLE CONTROL INPUT 2                  WO
INPT3       EQU X'0B'   PADDLE CONTROL INPUT 3                  WO
INPT4       EQU X'0C'   PLAYER 0 FIRE BUTTON INPUT              WO
INPT5       EQU X'0D'   PLAYER 1 FIRE BUTTON INPUT              WO

BACKGRND    EQU X'20'   BACKGROUND COLOR                        R/W
P0C1        EQU X'21'   PALETTE 0 - COLOR 1                     R/W
P0C2        EQU X'22'             - COLOR 2                     R/W
P0C3        EQU X'23'             - COLOR 3                     R/W
WSYNC       EQU X'24'   WAIT FOR SYNC                           STROBE
P1C1        EQU X'25'   PALETTE 1 - COLOR 1                     R/W
P1C2        EQU X'26'             - COLOR 2                     R/W
P1C3        EQU X'27'             - COLOR 3                     R/W
MSTAT       EQU X'28'   MARIA STATUS                            RO
P2C1        EQU X'29'   PALETTE 2 - COLOR 1                     R/W
P2C2        EQU X'2A'             - COLOR 2                     R/W
P2C3        EQU X'2B'             - COLOR 3                     R/W
DPPH        EQU X'2C'   DISPLAY LIST LIST POINT HIGH            WO
P3C1        EQU X'2D'   PALETTE 3 - COLOR 1                     R/W
P3C2        EQU X'2E'             - COLOR 2                     R/W
P3C3        EQU X'2F'             - COLOR 3                     R/W
DPPL        EQU X'30'   DISPLAY LIST LIST POINT LOW             WO
P4C1        EQU X'31'   PALETTE 4 - COLOR 1                     R/W
P4C2        EQU X'32'             - COLOR 2                     R/W
P4C3        EQU X'33'             - COLOR 3                     R/W
CHARBASE    EQU X'34'   CRARACTER BASE ADDRESS                  WO
P5C1        EQU X'35'   PALETTE 5 - COLOR 1                     R/W
P5C2        EQU X'36'             - COLOR 2                     R/W
P5C3        EQU X'37'             - COLOR 3                     R/W
OFFSET      EQU X'38'   FOR FUTURE EXPANSION - STORE ZERO HERE  R/W
P6C1        EQU X'39'   PALETTE 6 - COLOR 1                     R/W
P6C2        EQU X'3A'             - COLOR 2                     R/W
P6C3        EQU X'3B'             - COLOR 3                     R/W
CTRL        EQU X'3C'   MARIA CONTROL REGISTER                  WO
P7C1        EQU X'3D'   PALETTE 7 - COLOR 1                     R/W
P7C2        EQU X'3E'             - COLOR 2                     R/W
P7C3        EQU X'3F'             - COLOR 3                     R/W

SWCHA       EQU X'280'  P0, P1 JOYSTICK DIRECTIONAL INPUT       R/W
SWCHB       EQ0 X'282'  CONSOLE SWITCHES                        RO
CTLSWA      EQU X'28I'  I/O CONTROL FOR SWCHA                   R/W
CTLSWB      EQU X'283'  I/O CONTROL FOR SWCHB                   R/W
```

APPENDIX 3: DMA TIMING

There is some uncertainty as to the number of cycles DMA will require, because the internal MARIA chip timing resolution is 7.16 MHz, while the 6502 runs at either 1.79 MHz or 1.19 MHz. As a result, it is not known how many extra cycles will be needed in DMA startup/shutdown to make the 6502 happy. It is even possible for the 6502 to be in the middle of a long (TIA or 6532) access when it is to be halted, so the uncertainty goes up to about 5 cycles.

All times listed below refer to 7.16 MHz cycles.

DMA startup	5 - 9	cycles
Header (4 byte)	8	cycles
Header (5 byte)	12	cycles
Graphics Reads:		
Direct	3	cycles
Indirect/1 byte	6	cycles
Indirect/2 byte	9	cycles
Character Map access	3	cycles
Shutdown Times:		
Last line of zone	10 - 13	cycles
Other lines in zone	4 -7	cycles

End of VBLANK is made up of a DMA startup plus a Long shutdown.

DMA does not begin until 7 cpu (1.79 MHz) cycles into each scan line. The significance of this is that there is enough time to change a color, or change CTRL before DMA begins, and during HBLANK (before display begins). This figure should, however, be included in any DMA useage calculations.

Another timing characteristic is that there is one mpu (7.16 MHz) cycle between DMA shutdown and generation of a DLI.

APPENDIX 4: FRAME TIMING